It's Not Just for Memory Anymore

An Introduction to PCMCIA

You know the credit-card revolution has made a significant impact when you look at laptop peripherals today. Lalo's article introduces us to this evolving breed of devices.

FEATURE ARTICLE

Lalo J. Gastriani

It was around the beginning of the '80s when a few curious musicians started wondering about those "funny little connectors" found on the back panel of high-end synthesizers and sampling keyboards. Accessory catalogs were studied; inquiries were made.

Soon, they learned that expansion cards could be obtained and plugged into those sockets. By issuing the correct commands, keyboard parameter sets or "patches" as well as waveform data could be saved to the small memory cards and recalled later. The fact that these cards could be inserted or extracted while the unit was running made them even more attractive.

The offspring of those funny little sockets have now found their way onto virtually every notebook computer made today. The devices, which plug into these sockets, grew out of the JEIDA (Japan Electronics Industry Development Association) standard and are now known as *personal computer cards* or simply *PC cards.* These credit-card-sized peripherals contain anything from real-time global positioning systems to tiny hard drives with rotating media. Even more prevalent now are high-speed modems, which enable people on the road to "jack in" to that Infobahn we keep hearing about.

The computer industry is just beginning to see the benefits and potential of PC cards. For about the last three years, the standard for these devices and the system software that controls them, collectively referred to as PCMCIA (Personal Computer Memory Card International Association), have quickly become more known to both manufacturers of computers and end-users alike. As it now rolls into its latest 3.0 incarnation, the specification has endured the test of time and public scrutiny.

After a slow start, great strides have been taken in discovering the essential criteria which enable the cards to operate on a particular host and also provide the foundation for successful real-world, cross-platform operation. Although we're not there yet, this achievement will provide us with what is known in the industry as *interoperability.*

Already, we are at the point that a notebook computer is considered stripped down if it doesn't contain at least one PCMCIA socket. What's more, PCMCIA cards are sold everywhere from giant, computer-warehouse stores in your neighborhood to the shop-at-home cable channels. Consumers are asking manufacturers of desktop systems to provide PC-card capability as part of their solution on subnotebooks as well as desktops. Perhaps, we may soon see sockets on everything from automobiles to public telephones.

In the embedded world, PCMCIA sockets and cards offer a whole new breed of device. Designers can simply add a socket or two to their designs along with the proper enabling firmware, and flexibility and future expansion is possible.

But, let's take a closer look at PCMCIA.

HARDWARE

Sockets are the basic receptacle PC cards are inserted into and removed from. The socket connector itself normally rests inside the host computer and consists of a plastic housing with a double row of 34 pins giving a total of 68 pins, each spaced 1 mm apart. Photo 1 and Figure 1 illustrate the pin layout in a PCMCIA card end. The exact pinout of the socket can be found in 4.2–4.3 of *PCMCIA PC Card Standard Release 2.01.*

Type	Thickness (mm)
I	3.3
II	5
III	10.5
IV	13.5

Table 1—*Since different functions require different amounts of physical space, four card thicknesses are defined. Type IV is a nonstandard form factor, which PCMCIA rejected but still is used by some manufacturers.*

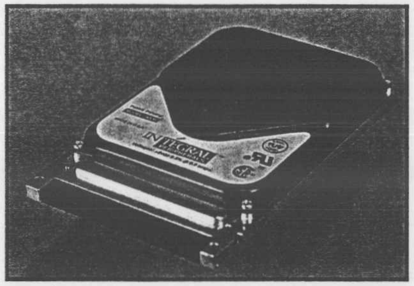

Photo 1—*A Type III PCMCIA card typically houses the rotating media of a hard drive.*

Cards are inserted into and extracted from the sockets while the host's system power is active, a technique also known as *hot swapping.* Hot swapping and the whole notion of temporal devices in personal computers is a fairly new concept and as such has its own unique set of problems. These problems center mainly on the fact that, until now, system resources were allocated to devices present at system bootstrap. The devices stayed present throughout the duration of the computing session, remaining nearly entirely static until the session was terminated.

Plugging a PC card into a powered-up system and expecting the functionality of the card to suddenly become active is quite an expectation. How this is done is where the real magic of technology lies.

If you were to look crosswise at the socket pins, you'd see that some of the pins do not protrude as far as others. The power rails are located on the outermost pins and are the longest of the pins. When a card is inserted, they make contact first. Similarly, when it is extracted, they retain the longest. This power arrangement enables buses to be powered up and tristated when (or very soon after) the card is inserted.

In addition to data and address lines, there are also control lines, battery indicators, a single interrupt line, and a card-detect line, which can indicate card insertions and extractions.

To be able to insert a modem card, have the OS recognize and configure it, spawn a communication program, connect to an on-line service, download E-mail, and then disconnect is an amazing series of events. To make the scenario even more interesting, remove the modem card and insert a 120-MB, rotating hard drive containing a customer database, which launches an invoice program.

The interface between the socket's 68-pin bus and the host is known as the *socket controller.* It manages the low-level aspects of the socket (i.e., power, interrupt routing, memory and I/O window allocation, etc.) according to program control.

On desktop and notebook systems, the socket controller is typically a dedicated chip which is part of some

Memory Only Card Interface: Always available at card insertion

Pin	Signal	I/O	Function
1	GND		Ground
2	D3	I/O	Data bit 3
3	D4	I/O	Data bit 4
4	D5	I/O	Data bit 5
5	D6	I/O	Data bit 6
6	D7	I/O	Data bit 7
7	$\overline{CE1}$	I	Card enable
8	A10	I	Address bit 10
9	$\overline{OE}$	I	Output enable
10	A11	I	Address bit 11
11	A9	I	Address bit 9
12	A8	I	Address bit 8
13	A13	I	Address bit 13
14	A14	I	Address bit 14
15	$\overline{WE/PGM}$	I	Write enable
16	RDY/$\overline{BSY}$	O	Ready/Busy
17	V_{CC}		
18	$V_{PP}1$		Programming Supply Voltage 1
19	A16	I	Address bit 16
20	A15	I	Address bit 15
21	A12	I	Address bit 12
22	A7	I	Address bit 7
23	A6	I	Address bit 6
24	A5	I	Address bit 5
25	A4	I	Address bit 4
26	A3	I	Address bit 3
27	A2	I	Address bit 2
28	A1	I	Address bit 1
29	A0	I	Address bit 0
30	D0	I/O	Data bit 0
31	D1	I/O	Data bit 1
32	D2	I/O	Data bit 2
33	WP	O	Write protect
34	GND		Ground

Memory Only Card Interface: Always available at card insertion

Pin	Signal	I/O	Function
35	GND		Ground
36	$\overline{CD1}$	O	Card detect
37	D11	I/O	Data bit 11
38	D12	I/O	Data bit 12
39	D13	I/O	Data bit 13
40	D14	I/O	Data bit 14
41	D15	I/O	Data bit 15
42	$\overline{CE2}$	I	Card enable
43	RFSH	I	Refresh
44	RFU		Reserved
45	RFU		Reserved
46	A17	I	Address bit 17
47	A18	I	Address bit 18
48	A19	I	Address bit 19
49	A20	I	Address bit 20
50	A21	I	Address bit 21
51	V_{CC}		
52	$V_{PP}2$		Programming Supply Voltage 2
53	A22	I	Address bit 22
54	A23	I	Address bit 23
55	A24	I	Address bit 24
56	A25	I	Address bit 25
57	RFU		Reserved
58	RESET	I	Card reset
59	$\overline{WAIT}$	O	Extend bus cycle
60	RFU		Reserved
61	$\overline{REG}$	I	Register select
62	BVD2	O	Battery voltage detect 2
63	BVD1	O	Battery voltage detect 1
64	D8	I/O	Data bit 8
65	D9	I/O	Data bit 9
66	D10	I/O	Data bit 10
67	CD2	O	Card detect
68	GND		Ground

I/O and Memory Card Interface: Available only after socket has been put into I/O mode

Pin	Signal	I/O	Function
16	IREQ	O	Interrupt request
18	$\overline{Vpp1}$		Programming & Peripheral Supply
33	IOIS16	O	IO Port is 16-bit
44	$\overline{IORD}$	I	IO Read
45	$\overline{IOWR}$	I	IO Write
52	Vpp2		Programming & Peripheral Supply 2
60	$\overline{INPACK}$	O	Input Port Acknowledge
61	$\overline{REG}$	I	Register select and IO Enable
62	$\overline{SPKR}$	O	Audio Digital Waveform
63	$\overline{STSCHG}$	O	Card Statuses Changed

Figure 1—*The 68-pin PCMCIA interface between a PC card and its socket shows the signal definitions in memory mode. The lower table describes those pins whose functions change when the socket is in I/O mode. The mode is determined by the state of the *REG signal.*

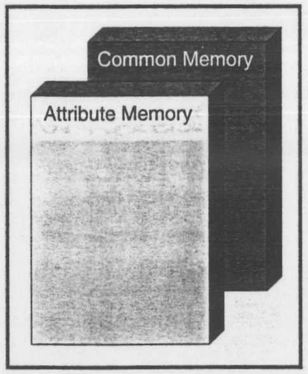

Figure 2—*Common and attribute memory share the same address space. The client selects which one is active.*

sort of host adapter. The Intel 82365SL and the Databook 86082 are examples of popular socket controllers.

In embedded systems, the socket controller can be implemented with discrete logic, as one of the standard pieces of silicon, or as a single task running along with other tasks in the microprocessor. As another possible route, you could use a dedicated microcontroller like the PIC.

RESOURCES

As I mentioned, the key to enabling PC cards in a system is to provide them the resources they require. However, before I talk about how a card tells the system what resources it needs, let's take a look at the various categories of resources available on a host.

Power is the most obvious and most basic resource. It is provided to the card via the power rails and is typically 3.3 or 5 V, depending on the host system. A typical socket controller may be requested to provide higher voltages for certain cards (e.g., flash-memory cards, which usually require 12 V for programming). To switch the power rails to the card, the socket controller either has built-in active switches or external MOSFET devices.

To interface a particular card's function to the rest of the system, memory windows are created which map a particular region of the card's address space into the host system's address space. The exact place within the host map is determined by system software. The size of the memory window as well as its attributes are usually also programmable.

On systems with I/O space, a card's I/O decoding might require connection to a particular range of I/O locations. This is particularly critical when a PC card is set up to emulate a static device.

Another important resource sometimes applicable is the PC card's interrupt routing. Refer again to Figure 1. Notice that a card has only a single IREQ line which can be routed through the socket controller to any host interrupt line. This routing is usually programmable, allowing for flexible compatibility.

The IREQ signal from the card should not be confused with the card-detect interrupt from the socket, which is used for insertion and extraction detection and has nothing to do with the card's IREQ signal. Instead, the IREQ line is to be used by the card in a card-specific manner.

While the currently released PCMCIA specs do not include support for DMA to and from a PC card, the next release (version 3.0) accommodates it. Socket controllers are already appearing which have the ability to provide DMA channels to and from the card's address space. Until the DMA functionality of the PCMCIA software layers are standardized, DMA implementations are ad hoc.

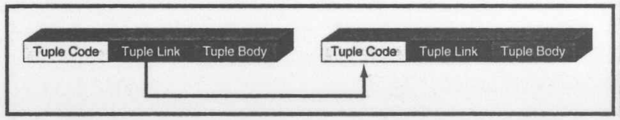

Figure 3—*Every card has information built into it detailing its manufacturer, part number, and other items. The information is conveyed to the host through the use of tuples, which form the basis for a linked list of data objects.*

CARD FORM FACTORS

There are currently three types or configurations of PC cards recognized by the standards: I, II, and III. The primary difference between the types is the thickness of the card.

The Japanese consortium of card standards (JEIDA) has taken a small lead in this area having designated a fourth type of card, which they call *Type IV*. Although the Type-IV form factor was rejected by the PCMCIA committee, the 13.5-mm slot can be found on some Toshiba models. Table 1 shows the correlation between thickness and type.

MEMORY SPACE

As Figure 2 illustrates, there are two types of memory space on a PC card: common and attribute. Even I/O cards must implement attribute memory for this is where the card's CIS (card information structure) is. The CIS contains the card's "biography" as well as its "wish list" for system resources.

By definition, when a card is powered up, it initializes in memory mode. This ensures that as each card is inserted, it begins in a known state. This state can best be characterized as:

- memory mode active (as opposed to I/O mode active)
- attribute memory active (as opposed to common memory active)

Attribute and common memory share the same address space (64 MB), but only one can be active at a time. The socket controller provides the means to select one of the two memory spaces.

Attribute memory is typically (although not always) nonvolatile, contains the CIS, and if applicable, the configuration registers. Attribute memory must begin at offset 0, but need not be in a single contiguous region. Most cards implement their CIS and configuration registers at fairly low offsets in attribute space (according to PCMCIA guideline), which ensures addressability by all hosts. This also minimizes that amount of page mapping required by the system software to access these regions.

A quirk of attribute space is that it is a 16-bit interface only. In other words, when reading CIS information or writing configuration registers, only the even bytes are considered valid data. Likewise, on write cycles, the card is only obliged to handle writes to the even bytes.

Mass storage PC cards have a separate memory space called *common memory*, which provides the main write/read memory space for the card. In a 4-MB SRAM card, for instance, common memory is viewed as a 4-MB linear region. By setting up a mapping window and by manipulating the offset addressed by this window into common memory, any byte contained on the card can be accessed.

Common memory is treated as an 8-bit array of bytes. Both the even and odd bytes are valid.

CARD INFORMATION STRUCTURE (CIS)

The Card Information Structure is implemented in a format known as the *metaformat*. As mentioned, the CIS is the card's biography, which contains, among other things, the:

- manufacturer
- part number
- voltage and current requirements
- absolute maximum ratings
- one or more configuration scenarios (I/O cards only)

In theory, when a card is inserted, system software powers up the card and starts looking at the CIS. As the CIS is traversed and parsed, more information about the card may be obtained.

This information is conveyed to the client through the use of *tuples*. Tuples form the basis for a linked list of data objects which provide enough information about the card that it may be properly enabled and used. Figure 3 presents the layout of a tuple.

As you can see, layer 1 offers the basic compatibility tuples. The first byte contains the tuple code and may fall into the range of 00–1D (hex). The values between 01E and 3F are reserved for future expansion. Layer 2 is for data recording format and lies

between 40 and 45, Layer 3 provides data organization and is in the range of 46–7F, and Layer 4 offers system-specific features and is found in 80–FE. The special value FF present as either a tuple code or a link marks the end of the tuple list. Table 5.2 of section 5.7 of the PCMCIA *PC Card Standard* describes all tuple codes and their meaning.

This scheme is quite flexible and allows for simple or complex CIS structure. Special tuples (long links) allow the CIS to span attribute and common memory space, thereby enabling elaborate CIS schemes to be implemented.

CARD FUNCTIONS

Memory cards are the simplest of PC cards to configure. They typically require only power for reading, writing, and mapping a window. In the case of nonvolatile memory like flash, special programming of voltage and current requirements can be found in the card's CIS.

I/O cards are the most prevalent of PC cards. By far the most popular I/O card is the modem or fax/modem card. I/O cards may or may not require I/O windows to be set up for them through the socket controller. When I/O is needed, only certain I/O locations are allowed to be decoded by the card.

This technique enables a PC card to look like any device that would normally be attached to a particular host's bus. By setting up the proper I/O windowing, the I/O card's register set can be made to respond to host bus requests within certain ranges.

Some I/O cards have memory-mapped registers, which are present in either the card's common or attribute memory space. By supporting this kind of operation, a particular card can ensure interoperability (or at least a higher chance of achieving it) by working in systems that have I/O space (e.g., the Intel x86 series) or those that do not (e.g., the Motorola 68k series).

All cards, when first powered up, assume memory mode. This facilitates the reading and parsing of the card's CIS. The client selects a particular configuration based on information in the CIS and a configuration scenario which makes sense for the host at that particular time. After configuration is selected and set, an I/O card then assumes I/O mode. The changeover at the socket level between memory and I/O mode causes a few pins to be redefined. Consult Figure 1 for the differences in pin definitions between memory and I/O mode.

Lately, some of the newer cards coming out have two or more separate and distinct functions. For example,

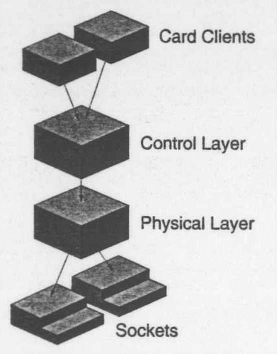

Figure 4—PCMCIA software is broken into several layers which apply mostly to desktop and notebook systems. By using separate layers, it's possible to have a common interface at the top and very different physical devices at the bottom.

Xircom has developed an Ethernet/fax-modem card, which combines a miniature Ethernet adapter and a high-speed fax/modem. Through this type of arrangement, both of the card's functions share the same socket.

This multifunctionality poses some challenges in the area of interrupt routing since there is only one IREQ pin from the card. This one IREQ pin, which is routed to one of the host's interrupt lines, must be able to signify an interrupt from two or more separate sources.

Some new proposals are being discussed by PCMCIA which include the use of virtual sockets to "house" the second and subsequent functions on a particular card. However, at this time, there have been no ratified extensions or enhancements to the specifications for multifunction cards.

Still another class of PC cards are those that appear to the system to be a standard peripheral known as an *AT attachment* or an *ATA device*. These cards may contain solid-state memory or rotating media. When the card's resources are properly configured, the card appears as a hard drive.

In a typical desktop or notebook application, these devices are connected to the host file system by a device driver and appear as another logical device. In an embedded system, an ATA device could contain the operating system (with a virtual memory-storage partition) and/or extra storage.

Many ATA devices contain programmable watchdogs. These enable the power management aspects of a particular OS to set timeout values for spinning down the drive during periods of inactivity.

SOFTWARE

Whether the environment is desktop, notebook, or embedded, the general intent of PCMCIA software at large is universal:

- enable cards within sockets
- protect systems from bad cards when possible
- encapsulate the details of socket electronics
- allow many clients of sockets to coexist
- manage resources
- provide interoperability and compatibility

How these goals are best achieved is almost entirely platform specific. But, the basic tenets are the same. By breaking up the software into layers and by defining each layer's boundaries and the part that layer serves greatly simplifies the conceptual visualization of implementing solutions.

PCMCIA software is broken up into two basic areas: clients and servers. The server layers provide the services to the socket controller. Clients, on the other hand, interact with the servers and other clients in an effort to configure the cards and keep them running.

Layers, outlined in the PCMCIA documents under socket services and card services, apply mostly to desktop and notebook systems. Figure 4 presents the various layers which make up PCMCIA software.

The physical layer controls the hardware registers or other programming model of the socket controller(s) on a particular platform. In essence, this layer reports the capabilities of each socket controller and enables the encapsulated manipulation of them.

A simple interface to higher-level layers allows these higher layers to deal with logical sockets. The PCMCIA standard calls this layer *socket services*. There may be one or more of them in a system, each handling one or more socket controller(s).

Above the physical layer lies the control layer which reacts to events, manages resources, and interacts with the card clients in a system. This layer is part of the operating system and contains further abstractions that, at this level, enable memory technologies to be encapsulated according to their topology and power requirements.

Each client makes resource requests to this layer. Each request is weighed against collisions with other system components; collisions cause rejections. This layer is called *card services* in PCMCIA's standards. There is only one case of it in any system.

Clients are tasks at the top of the PCMCIA food chain. When a card is inserted, the ensuing interrupt is handled by the physical and control layer. This eventually causes an insertion event or message to be broadcast to all clients registered with the control layer.

At this point, a client typically parses the card's CIS looking for either recognizable "landmarks" within the manufacturer's ID tuple or moving right into the configuration tuples. Once it starts configuration, it constructs resource requests and passes them off to the control layer.

This process continues until a particular configuration scenario is accepted by the control layer, at which time it is latched and the card is enabled. The card remains in this state until it is either extracted or some sort of power-management message is broadcast, forcing the card and socket into a lower-power state.

Extracting the card reverses the process so that resources are freed and given back to the host.

CONCLUSION

PCMCIA technology opens doors in many areas of desktop, notebook, and hand-held computing. This introduction provides a foundation for further study. In a follow-up article, I will focus more intensely on embedded systems, detailing an actual application of PCMCIA. ■

Lalo J. Gastriani is a consulting engineer based in Southern California. Since 1982, he has concentrated on real-time software architecture and design. Since 1990, he has been involved in PCMCIA. Lalo may be reached at ljg@kaiwan.com or P.O. Box 80801, Rancho Santa Margarita, CA 92688.

CONTACTS

PCMCIA specifications:
Personal Computer Memory Card International Assoc. (PCMCIA)
1030G East Duane Ave.
Sunnyvale, CA 94086
(408) 720-0107
Fax: (408) 720-9416

PCMCIA peripherals:
Integral Peripherals, Inc.
5775 Flatiron Parkway, Ste. 100
Boulder, CO 80301
(303) 449-8009
Fax: (303) 449-8089

Xircom
26025 Mureau Rd.
Calabasas, CA 91302
(818) 878-6423
Fax: (818) 878-7630

IRS

407 Very Useful
408 Moderately Useful
409 Not Useful